

ASSISTANT COMMISSIONER FOR PATENTS
BOX PATENT APPLICATION
WASHINGTON, D.C. 20231

3813 U.S. PTO



10/31/00

CASE DOCKET NO: YOR920000757US1
DATE October 31, 2000

3813 U.S. PTO
09/702927
10/31/00

Sir:

Transmitted herewith for filing under rule 1.53(f) is the Patent Application of:

Inventors: Seraphin B. Calo, Richard O. LaMaire, Ashish Mehra,
Anees A. Shaikh, Renu Tewari and Dinesh Verma.

For: Method and Apparatus for Distributed Application Acceleration

Enclosed are:

☒ 8 Sheets of Formal Drawings.

☐ An assignment of the invention to International Business Machines Corporation, Armonk, New York 10504.

☐ A certified copy of a _____ application number _____.

☐ Declaration and Power of Attorney is attached to the application.

☐ Associate Power of Attorney.

☐ Information Disclosure Statement with form PTO-1449 with references attached.

The filing fee has been calculated as shown below:

| | (Col. 1) | (Col. 2) |
|---|-----------|-----------|
| FOR: | NO. FILED | NO. EXTRA |
| BASIC FEE | | |
| TOTAL CLAIMS | 34 - 20 = | 14 |
| INDEP CLAIMS | 5 - 3 = | 2 |
| ____ MULTIPLE DEPENDENT CLAIM PRESENTED | | |

OTHER THAN A
SMALL ENTITY

| RATE | FEE |
|-----------|-------------|
| | \$ 710.00 |
| X \$ 18 = | \$ 252.00 |
| X \$ 80 = | \$ 160.00 |
| + \$270 = | \$ 0.00 |
| TOTAL | \$ 1,122.00 |

If the difference in Col. 1 is less than zero, enter "0" in Col. 2.

☒ Please charge my Deposit Account No. 09-0468 in the amount of \$1,122.00.

☒ The Commissioner is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit Account No. 09-0468. A duplicate copy of this sheet is enclosed.

☒ Any additional filing fees required under 37 CFR 1.16.

☒ Any patent application processing fees under 35 CFR 1.17.

Respectfully submitted,

By

Louis Herzberg

Louis P. Herzberg
Registration No.: 41,500
Tel. (914) 945-2885

IBM CORPORATION
INTELLECTUAL PROPERTY LAW DEPT.
P.O. BOX 216
YORKTOWN HEIGHTS, NY 10598

11-01-00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: S. B. Calo et al.

Docket No.: YOR920000757US1

Serial No.:

Group No.:

Filed: Herewith

Examiner:

For: METHOD AND APPARATUS FOR DISTRIBUTED
APPLICATION ACCELERATION

Assistant Commissioner for Patents
Washington, D.C. 20231

EXPRESS MAIL CERTIFICATE

Express Mail Label Number EL549238284US

Date of Deposit October 31, 2000

I hereby certify that the attached paper or fee

Patent Application Transmittal Letter (original and one copy)
Patent Application Under Rule 1.53(f) (Missing Parts)
Drawings (8 Sheets)
Return Postcard

is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

Jennifer A. Smith

(Name)


Signature

Note: Each paper must have its own certificate and the "Express Mail" label number as a part thereof or attached thereto. When, as here, the certification is presented on a separate sheet, that sheet must (1) be signed and (2) fully identify and be securely attached to the paper or fee it accompanies. Identification should include the serial number and filing date of the application as well as the type of paper being filed, e.g. complete application, specification and drawings, responses to rejection or refusal, notice of appeal, etc. If the serial number of the application is not known, the identification should include at least the name of the inventor(s) and the title of the invention.

Note: The label number need not be placed on each page. It should, however, be placed on the first page of each separate document, such as, a new application, amendment, assignment, and transmittal letter for a fee, along with the certificate of mailing by "Express Mail". Although the label number may be on checks, such a practice is not required. In order not to deface formal drawings it is suggested that the label number be placed on the back of each formal drawing or the drawings be accompanied by a set of informal drawings on which the label number is placed.

Method and Apparatus for Distributed Application
Acceleration

FIELD OF THE INVENTION

The present application relates to the field of computer networks and distributed applications. It is more particularly directed to applications that are operational on the Internet using a system of Web-browsers and Web-servers.

BACKGROUND OF THE INVENTION

Currently, access over the Internet to Web-based applications is provided by having a Web-browser connect directly over a network of routers to a Web-server that maintains static content in data files, and composes dynamic content by executing programs, typically cgi-bin scripts or Java servlets. However, during periods of congestion due to traffic patterns on the Internet, this arrangement results in poor response times for the end client. The situation is typically worse the farther the client is located from the Web-server and the greater the number of intermediary routers involved in the network connection.

One way to improve application response time, reliability, and availability is to distribute the applications to proxy servers located closer to the client browsers. Distribution of content is used to improve the performance of the network by means of proxies within the network that cache pages.

09:02:40
DOCKET NUMBER: YOR920000757US1

1 The simple caching approach works well for data that is
2 static and unchanging, e.g. images, video clips, etc. A
3 proxy server is deployed within the network in many
4 different ways. Some of the common ways include using a
5 proxy server as a reverse proxy, where the proxy server is
6 located closer to the web-server it is proxying for; as a
7 forward proxy, where the proxy server is located closer to
8 the browser or the client applications; or is as other
9 auxiliary servers which may be located elsewhere within the
10 network. The proxy server usually provides information to
11 the browser on behalf of a backend server. The browser may
12 contact the proxy server due to a variety of reasons e.g.,
13 because it has been explicitly configured to do so, or
14 because the domain name server gives it the location of a
15 proxy server instead of the backend web-server, or because a
16 network operator or backend web-server operator has
17 configured the network to send requests from the browser to
18 the proxy server in a transparent fashion.

19 However, the techniques of caching that are commonly
20 deployed in the current Internet do not work well with a
21 large portion of web-accessible content. Data that is
22 personalized to a client, or data that is generated by
23 invocation of programs like cgi-bin scripts or servlets can
24 not be readily cached at the proxies. For a server offering
25 electronic services over the Internet, non-static data forms
26 a significant portion of their overall data content. It
27 would be advantageous to have a scheme whereby such
28 dynamically generated content, and web-centric applications
29 can also benefit from the presence of proxies.

1 As in the case of caching of static data, it is highly
2 desirable that the caching of applications be done so that
3 the administrative and operation control of the
4 data/application resides with the original server, rather
5 than with at the proxy server. A solution is needed which
6 accelerates applications while still providing the
7 administrative control of the application to the original
8 server, rather than the proxy server.

9 SUMMARY OF THE INVENTION

10 Accordingly, an aspect of the present invention presents
11 methods and apparatus by which to accelerate execution of
12 Web front-ended applications by means of executing them at
13 proxies located closer to the client browsers.

14 Another aspect of the present invention presents methods and
15 apparatus for a proxy server which provides an execution
16 environment for acceleration of Web front-ended
17 applications.

18 Another aspect of the present invention presents methods and
19 apparatus for a backend server which provides an execution
20 environment for acceleration of Web front-ended
21 applications.

22 BRIEF DESCRIPTION OF THE DRAWINGS

23 These and other aspects, objects, features, and advantages
24 of the present invention will become apparent upon further

consideration of the following detailed description of the invention when read in conjunction with the drawing figures, in which:

Fig. 1 is a block diagram showing an example of an application acceleration infrastructure, and their manner of interaction in accordance with the present invention;

Fig. 2 is a block diagram of an example showing steps for a method of application acceleration in accordance with the present invention;

Fig. 3 is a block diagram of an example showing steps taken by a proxy server in executing a requested service for an end client in accordance with the present invention;

Fig. 4 is a block diagram of an example showing steps for determination of an appropriate set of programs to run in response to a particular end client service request in accordance with the present invention;

Fig. 5 is a block diagram of an example showing structure of an information record, used in a solution to the distributed application acceleration problem in accordance with the present invention;

Fig. 6 is a block diagram showing an example of a structure of a distributed system to implement a solution to the distributed application acceleration problem in accordance with the present invention;

1 Fig. 7 is a block diagram showing an example of a structure
2 of a proxy server, used in a solution to the distributed
3 application acceleration problem in accordance with the
4 present invention; and,

5 Fig. 8 is a block diagram showing an example of a structure
6 of a backend server, used in a solution to the distributed
7 application acceleration problem in accordance with the
8 present invention.

9 DETAILED DESCRIPTION OF THE INVENTION

10 Figure 1 shows an example of the components of an
11 application acceleration infrastructure. A communication
12 network 101 is used to interconnect client devices
13 containing Web-browsers like 102 and 103 to a backend web
14 server 104 that provides Web-based applications. Major
15 portions of the application code are moved onto proxy
16 servers like 105 and 106 that are in closer proximity to the
17 respective end client devices. Communications is maintained
18 between the proxy servers 105 and 106 and the backend server
19 104 so that the backend server can continue to exercise
20 administrative control over the distributed portions of the
21 applications code. Communications is also maintained
22 between the end client devices 102 and 103 and the backend
23 server 104 so that the latter can continue to provide
24 applications services that are not readily distributable.

25 Fig. 2 delineates the steps that comprise a method of
26 application acceleration. In response to a client service
27 request 201, a Wide Area Load Balancing module determines

1 the appropriate proxy server to which to direct the request
2 based upon current network performance characteristics and
3 the location of the end client device as in 202. The
4 proxy server then determines the set of programs that it
5 needs to run in order to satisfy the end client request as
6 in 203. These may be already resident at the proxy server
7 if they had been needed to satisfy previous requests from
8 other users, or if they had been deployed to the proxy
9 server in anticipation of end client requests for basic or
10 popular applications. In any of these cases, they are
11 resident at the backend server. The proxy server obtains
12 the data and application code appropriate for fulfilling the
13 service request as in 204, and executes the indicated
14 functions for the end client on behalf of the backend server
15 as in 205.

16 Fig. 3 is a block diagram that shows an example of steps
17 taken by a proxy server in executing a requested service for
18 an end client. The response from an application is often
19 tailored to a particular end client or class of end users,
20 depending upon organizational affiliations or other
21 criteria. The proxy server thus obtains information from
22 the backend server regarding which execution parameters to
23 employ in satisfying this specific end client request as in
24 301. The application logic is then executed as in 302, and
25 any administrative information or error messages resulting
26 from the application execution are logged and sent to the
27 backend server as in 303. In order to simplify systems
28 management and provide support for problem tracking and
29 diagnosis, the backend server maintains responsibility for
30 the application logic. This allows the proxies to be shared
31 effectively by many back end servers without increasing

1 their complexity. The results of the execution of the
2 application logic are then sent to the end client as in 304.

3 Fig. 4 is a block diagram of steps for a determination of
4 an appropriate set of programs to run in response to a
5 particular end client service request. The request
6 indicates a target location within the backend server for
7 the application logic, typically in the form of a Uniform
8 Resource Locator (URL). The proxy server maintains data on
9 application programs that it represents, and determines what
10 service is being requested as in 401. The appropriate
11 proxylet record is then retrieved as in 402. The proxylet
12 record contains a data field specifying the set of programs
13 needed in order to properly satisfy an end client service
14 request, and this is read by the proxy server as in 403.

15 Figure 5 shows the structure of a proxylet record used in
16 order to determine the set of programs that are to be
17 executed at a proxy when a request is received from a
18 client. The proxylet record 501 includes several fields.
19 The RequestedURL field 503 contains the URL which the client
20 was requesting. The ExecuteURL field 505 contains the name
21 of a URL which will be executed at the local proxy in order
22 to support this request. The location where the compute
23 code required to run ExecuteURL is contained in the field
24 CodeLocation 507. The ParameterList field 509 contains any
25 parameters which would be passed to the program in order for
26 it to be executed. The contents of the ParameterList field
27 509 may be same for different proxies within the network, or
28 they may be different for multiple proxy servers. The
29 ExpirationTime field 511 contains the date until which this
30 proxylet record may be considered valid. Once the time

1 specified by field 511 has expired, a proxy server would
2 need to retrieve a new proxylet-record from the backend
3 server. The LoggerURL field 513 identifies a location where
4 the error messages and diagnostic output of the proxylet are
5 provided. Typically, the loggerURL would identify a
6 location on the backend server. The codeVersion field 515
7 contains the time when the program set identified in the
8 field codeLocation 505 was last modified. Other fields may
9 also be included within the proxylet-record as used in
10 different embodiments of the invention.

11 As an example, let us consider a request from a client which
12 is targeted to the location

13 http://main-server.com/servlet/program1.

14 This request is delivered to the proxy server running at the
15 machine proxy-server.com. The proxylet record for this
16 request might contain the fields of requestedURL field being

17 http://main-server.com/servlet/program1,

18 with executeURL field being /servlet/proxy-program1, the
19 codeLocation field being

20 http://main-server.com/proxylet/prox-program1,

21 the ParameterList being empty, and the LoggerURL being

22 http://main-server.com/servlet/logger,

1 the expirationTime field being 30000 seconds after a
2 reference date such as 1/1/1970, and the codeVersion field
3 being 29500 seconds after a reference date such as 1/1/1970.

4 When such a request is received at the machine
5 proxy-server.com, the machine checks if it has a cached
6 entry corresponding to the proxylet-record where the
7 requestedURL matches the request being received. If it
8 does, it checks the expirationTime field to ensure that the
9 record needs to be updated. It then checks to see if it has
10 the program identified by the executeURL installed locally
11 at the proxy-server. If it does, then it runs the program
12 passing to it any parameters contained in the parameterList
13 field. If the proxylet-record is not found, or if the
14 current time is greater than the time specified in the
15 expirationTime field, the proxy-server contacts the
16 main-server to obtain a fresh copy of the proxylet-record
17 prior to executing the steps outlined above.

18 The schemes as described above can be seen as a distributed
19 system that achieves acceleration of applications by
20 distributing their execution. An apparatus 601 that
21 implements the distributed system is shown in Figure 6.
22 Figure 6 includes three distributed components, a wide-area
23 load balancer 603, a application distributor 605, and a base
24 class library 607. The wide area load-balancer 605 is a
25 distributed module which collects performance statistics
26 about the network, and determines the most appropriate place
27 where a request should be sent out to. Since these
28 components are distributed components, the connectivity
29 information among them is not shown in the Figure. However,

1 the connectivity information will become clearer from the
2 description provided below.

3 The Wide Area Load Balancer 603 is a component responsible
4 for distributing client requests to different proxy servers
5 within the network. It can be implemented in a variety of
6 manners. One common way to implement it is by means of a
7 modified domain name server. The domain name server,
8 usually abbreviated to DNS server, is the application in the
9 network responsible for mapping machine names to IP
10 addresses. A modified domain name server can return an IP
11 address which corresponds to an appropriate proxy server
12 when a client requests an address for the backend server.
13 The appropriate proxy server is determined on the basis of
14 the current network performance characteristics and the
15 location of the client.

16 An alternative implementation of the wide area load balancer
17 includes a module within the backend server that is
18 responsible for redirecting requests to the appropriate
19 proxy server. Such a redirection module might be
20 implemented as a plug-in module among a variety of
21 web-servers such as Apache, NetScape or Microsoft IIS
22 server, which are commonly in use in the industry. The
23 module would look at a table of redirection rules, which
24 specify how requests coming from specific client IP
25 addresses should be dispatched, and use this information to
26 determine the appropriate proxy-server to which the request
27 should be dispatched. The selection of the proxy-server can
28 be based on other criteria included in the rule, e.g. The
29 resource (URL) being requested by the client, or a cookie
30 which is contained within the client's request.

Another embodiment of the wide area load balancer uses a stand-alone http server which provides the same functionality as that of the module described above. The http server implements the ability to direct requests to proxy-servers, or to another server operation locally at the site with the stand-alone http server, which services requests that need to be performed locally.

The application distributor 605 is responsible for ensuring that the set of programs that need to be executed at the proxy server are indeed available at that server. There are many embodiments of such an application distributor which will be useful to those skilled in the art.

One useful embodiment of an application distributor uses a program which keeps track of all the programs and data that is available at a main-server, and maintains a replica of those program and data at the proxy server. Such an distributor will push the changes that occur at the main-server out to the proxy servers in order to maintain this consistency of program and data.

Another embodiment of an application distributor uses a program that runs at the proxy-server and caches a copy of programs and data from the main-server when requests that would cause execution of those programs are received at the proxy-server. The data that the programs need to execute is also retrieved as needed, and cached at the proxy-server. The programs and data are both cached on demand.

1 Yet another embodiment of an application distributor uses a
2 program that employs both of these techniques. Some of the
3 programs which are most often used are pushed out to all of
4 the proxy-servers while other programs are cached on demand
5 at the proxy-server.

6 The class library 607 is a set of programs that exists at
7 all the proxy-servers and the backend servers. It contains
8 a collection of classes that enable many functions to occur.

9 One of the classes contained in the library identifies the
10 set of programs that are capable of executing at the
11 proxy-server. All such programs are derived from a specific
12 class proxylet, and the fact that these programs are
13 subclassed from the class proxylet is used to validate that
14 the program can execute at the proxy-server. Yet another
15 class provided in the class library is the Logger class,
16 which allows the output and error messages generated by the
17 program executing at the proxy-server to be copied to the
18 main-server for purposes of logging and diagnostics. Yet
19 other set of classes allow for the caching of different
20 types of application data. Instances of these include the
21 programs for caching queries made to a directory or a
22 database, programs for caching records in a database, as
23 well as programs for caching files.

24 The components of the distributed architecture shown in
25 Figure 6 are various proxy-servers and the backend server.
26 A proxy-server which is such a component in this solution is
27 shown in Figure 7. The proxy-server 701 includes a Cache
28 Manager 703, a set of cached information records 705, a set
29 of cached programs 707, and a set of cached data 709. The
30 Cache Manager 703 is responsible for managing and updating

the different types of caches, namely the set of cached information records 705, the set of cached programs 707 and the set of cached data 709. The cache manager maintains all of these caches in an appropriate manner. The set of cached information records 705 contains information about the cached programs that are cached locally in the set of cached programs 707. The format of information contained in the information record is similar to that of the proxylet record 501 shown in Figure 5. The Cache Manager 703 utilizes the services of an application distribution module 605 in order to ensure the consistency and coherence of the different sets of caches. The set of cached data programs 709 is maintained current by using data caching techniques that are well-known to those versed in the art.

Figure 8 shows the structure of a backend server which would respond to the proxy server shown in Figure 7 and provides the other part of the infrastructure for application acceleration. The backend server 801 includes a traditional web-server 803, a set of programs to be downloaded to proxy servers 805, a set of local programs 807, and a set of operational programs 809. The web-server 803 provides the means by which a proxy server can gain access to the set of programs 803, 805 and 807. The set of downloadable programs 803 are transferred to a proxy server if it makes a request for them. All or a subset of the programs may be transferred to the proxy server. The set of local programs 807 provides a means by which a proxy server can execute some parts of the processing at the backend server itself. As an example, a proxy server may want to execute programs related to updating databases only at the backend server. The set of operational programs 809 provides a means by

1 which a proxy-server can provide diagnostics and management
2 information to the backend server. An example of an
3 operational program 809 would be a logger servlet that can
4 obtain logging messages generated by programs executing at
5 the proxy server.

6 In some embodiments of the backend server, the web-server
7 may incorporate an ability to redirect client requests to
8 other servers. This would be an instance of the application
9 distribution module. In other embodiments, the backend
10 server may rely upon the domain name service to do such
11 redirections. The backend server as described in Figure 8
12 and a set of proxy servers as described in Figure 7 together
13 provide the infrastructure for distributed application
14 acceleration.

15 It is noted that the present invention can be realized in
16 hardware, software, or a combination of hardware and
17 software. A tool according to the present invention can be
18 realized in a centralized fashion in one computer system, or
19 in a distributed fashion where different elements are spread
20 across several interconnected computer systems. Any kind of
21 computer system - or other apparatus adapted for carrying
22 out the methods described herein - is suitable. A typical
23 combination of hardware and software uses a general purpose
24 computer system with a computer program that, when being
25 loaded and executed, controls the computer system such that
26 it carries out the methods described herein. The present
27 invention can also be embedded in a computer program
28 product, which comprises all the features enabling the
29 implementation of the methods described herein, and which -

DOCKET NUMBER: YOR920000757US1

1 when loaded in a computer system - is able to carry out
2 these methods.

3 Computer program means or computer program in the present
4 context includes any expression, in any language, code or
5 notation, of a set of instructions intended to cause a
6 system having an information processing capability to
7 perform a particular function either directly or after
8 either conversion to another language, code or notation;
9 and/or reproduction in a different material form.

10 It is noted that the foregoing has outlined some of the more
11 pertinent objects and embodiments of the present invention.
12 This invention may be used for many applications. Thus,
13 although the description is made for particular arrangements
14 and methods, the intent and concept of the invention is
15 suitable and applicable to other arrangements and
16 applications. It will be clear to those skilled in the art
17 that modifications to the disclosed embodiments can be
18 effected without departing from the spirit and scope of the
19 invention. The described embodiments ought to be construed
20 to be merely illustrative of some of the more prominent
21 features and applications of the invention. Other
22 beneficial results can be realized by applying the disclosed
23 invention in a different manner or modifying the invention
24 in ways known to those familiar with the art.

1 CLAIMS

2 Having thus described our invention, what we claim as new
3 and desire to secure by Letters Patent is as follows:

4 1. A method for distributing at least one application in a
5 communication network, said method comprising the steps of:

6 redirecting to one server of a plurality of proxy
7 servers at least one service request received from a
8 client for said at least one application;

9 determining a set of programs required at said one
10 server to fulfil said request for said at least one
11 application; and

12 executing said set of programs.

13 2. A method as recited in claim 1, further comprising
14 examining a cache of programs to obtain the set of programs;

15 3. A method as recited in claim 2, wherein said cache is
16 located at another server of said plurality of proxy
17 servers.

18 4. A method as recited in claim 2, further comprising
19 returning the results of the step of executing to the
20 client.

1 5. A method as recited in claim 1, further comprising
2 forwarding a portion of the request that needs to be
3 satisfied at another server to said another server.

4 6. A method as recited in claim 5, wherein said another
5 server is a backend server.

6 7. A program storage device readable by machine, tangibly
7 embodying a program of instructions executable by the
8 machine to perform method steps for distributing at least
9 one application in a communication network, said method
10 comprising the steps of claim 1.

11 8. A method as recited in claim 1, wherein the step of
12 executing includes:

13 obtaining parameters for execution from a backend
14 server; and

15 writing any resulting logging and error messages to
16 said backend server.

17 9. A method as recited in claim 1, where the step of
18 determining includes parsing the request to determine the
19 program required to satisfy the request.

20 10. A method as recited in claim 9, further comprising:

21 retrieving a proxylet-record for said program; and

```

looking up a field of said proxylet-record for
determining the set of programs to be executed at
the proxy server;

```

11. A method as recited in claim 2, where the step of examining includes:

employing a local store in determining a first set of programs present at the first proxy server; and

downloading a second set of programs from another server for said second set of programs not present at said proxy.

12. A method as recited in claim 1, where the step of redirecting is based upon a-priori knowledge of location of said set of programs.

13. A method as recited in claim 12, wherein said a-priori knowledge is deployed at a domain name server.

14. A method as recited in claim 12, wherein said a-priori knowledge is deployed at a backend server.

15. An apparatus to accelerate a distributed application within a network, the apparatus comprising:

a wide area load balancer for distributing at least one request from at least one client to a particular proxy server from among a plurality of proxy servers;

1 an application distributor for distributing a set
 2 of programs used for said distributed application
 3 to at least said particular proxy server necessary
 4 to satisfy said at least one request;

5 a information-management recorder for recording
 6 connectivity information about the set of
 7 programs; and

8 an execution device for executing said set of
 9 programs satisfying said at least one request at
 10 said particular proxy server.

11 16. An apparatus as recited in claim 15, further comprising
 12 a request forwarder for forwarding to another server any
 13 portions of said at least one request which have to be
 14 executed at said another server.

15 17. An apparatus for distributing at least one application
 16 in a communication network, comprising:

17 a proxy server having:

18 a set of programs used in said at least one
 19 application, said set of programs retrieved from a
 20 back-end server and executed locally to satisfy
 21 part of at least one request received from a
 22 client;

23 a set of cached data associated with said set of
 24 programs;

1 a set of information-management records for said
2 set of programs; and

3 a Cache Manager for maintaining the set of
4 programs, the set of cached data and the set of
5 information-management records in distribution of
6 said at least one application.

7 18. An apparatus for distributing at least one application
8 in a communication network, said apparatus comprising a
9 backend server having:

10 a first set of programs used for said at least one
11 application that, said set of programs being
12 distributed to at least one server of a plurality
13 of proxy servers within the network;

14 a second set of programs used for said at least
15 one application, said set of programs being
16 executed locally by the backend server;

17 a third set of programs used for said at least one
18 application, said third set of programs to receive
19 logging and error messages from the execution of
20 said first set of programs; and

21 an accessing server to provide access to the first
22 set of programs by any of the proxy servers.

23 19. An apparatus as described in claim 18, further
24 comprising a request redirector for redirecting requests to
25 one of the plurality of proxy servers.

001601 2620260

1 20. An article of manufacture comprising a computer usable
2 medium having computer readable program code means embodied
3 therein for causing application distribution in a network
4 with a plurality of machines, the computer readable program
5 code means in said article of manufacture comprising
6 computer readable program code means for causing a computer
7 to effect the steps of claim 1.

8 21. A method for distributing at least one application,
9 said method comprising:

10 redirecting one client for said at least one
11 application, to a first proxy server from a
12 plurality of proxy servers;

13 evaluating a request for said at least one
14 application to determine a part that is executable
15 at the first proxy server; and

16 executing said part at said proxy server.

17 22. A method as recited in claim 21, further comprising
18 obtaining at least one program used by said at least one
19 application enabling said step of executing.

20 23. A method as recited in claim 22, further comprising
21 determining a location of said at least one program.

22 24. A method as recited in claim 22, further comprising
23 obtaining values of parameters specific to said request

25. A method as recited on claim 24, wherein the step of
executing includes:

performing at least one operation to satisfy said
request; and

writing any resulting logging messages to a
backend server.

26. A method as recited in claim 25, wherein said backend
server is managing said at least one program.

27. A method as recited in claim 23, wherein said location
is the location of a second proxy server.

28. A method as recited in claim 23, wherein the step of
determining includes:

obtaining a proxylet-record for said request; and

looking up at least one field in the
proxylet-record.

29. A method as recited in claim 24, wherein the step of
obtaining includes:

obtaining a proxylet-record for said request; and

looking up at least one field in the
proxylet-record.

DOCKET NUMBER: YOR920000757US1

1 30. A method as recited in claim 21, further comprising
2 redirecting a second request from said first client to a
3 second proxy server.

4 31. A method as recited in claim 21, further comprising
5 redirecting a second request received from a second client
6 to said first proxy server.

7 32. A method as recited in claim 21, further comprising
8 redirecting a second request received from a second client
9 to a second proxy server.

10 33. An article of manufacture comprising a computer usable
11 medium having computer readable program code means embodied
12 therein for causing application distribution in a network
13 with a plurality of machines, the computer readable program
14 code means in said article of manufacture comprising
15 computer readable program code means for causing a computer
16 to effect the steps of claim 21.

17 34. A program storage device readable by machine, tangibly
18 embodying a program of instructions executable by the
19 machine to perform method steps for distributing at least
20 one application in a communication network, said method
21 comprising the steps of claim 21.

1 **Method and Apparatus for Distributed Application**
2 **Acceleration**
3 **ABSTRACT**

4 The present invention presents methods and apparatus
5 supporting acceleration of networked applications by means
6 of dynamic distributed execution and maintenance. It also
7 enables management and administration of the distributed
8 components of the networked applications from a responsible
9 point of origination. The method and apparatus deploys a
10 plurality of proxy servers within the network. Clients are
11 directed to one of the proxy servers using wide area load
12 balancing techniques. The proxy servers download programs
13 from backend servers and cache them in a local store. These
14 programs, in conjunction with data stored at cached servers,
15 are used to execute applications at the proxy server,
16 eliminating the need for a client to communicate to a
17 backend server to execute a networked application.

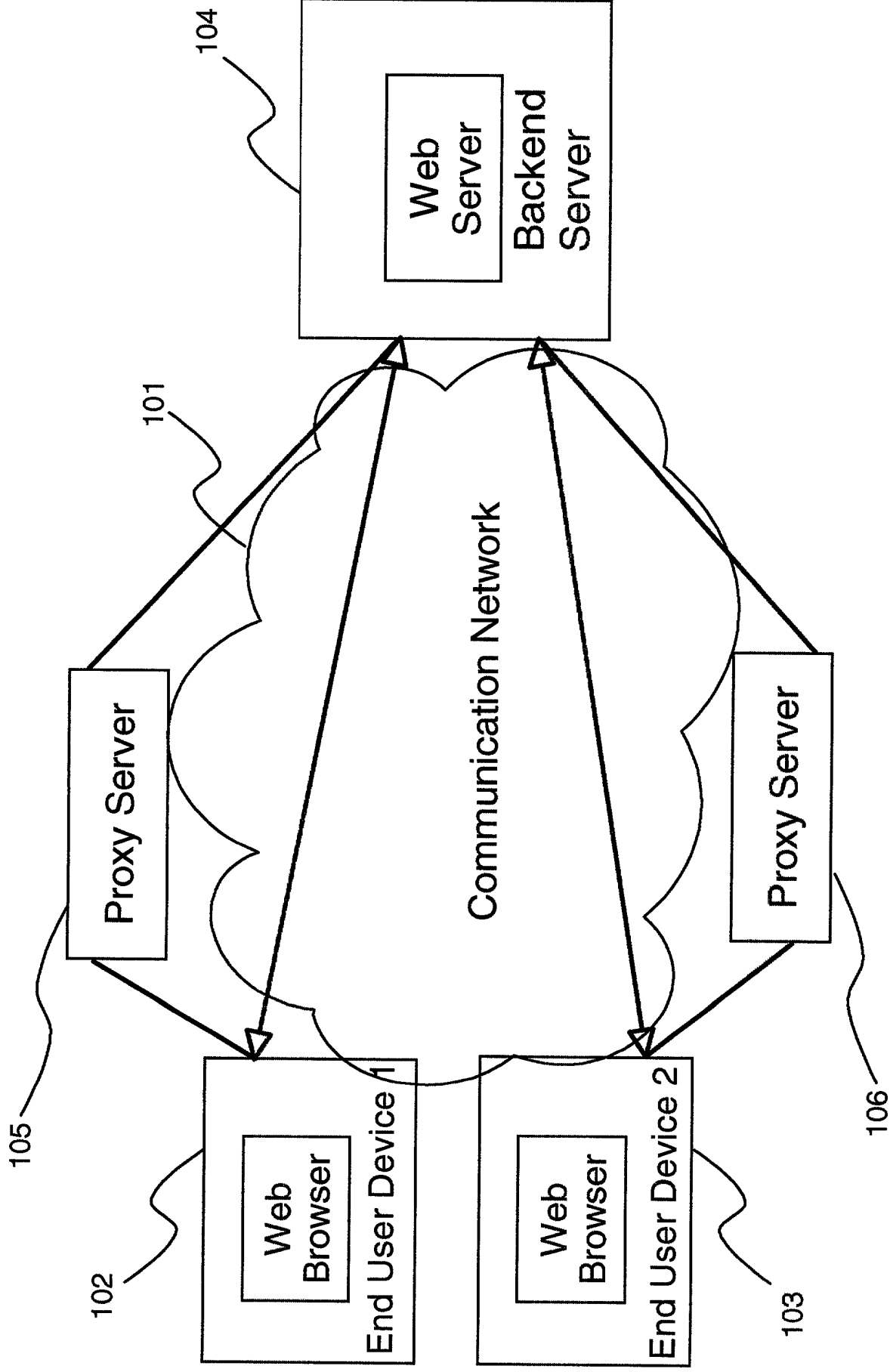


FIG 1

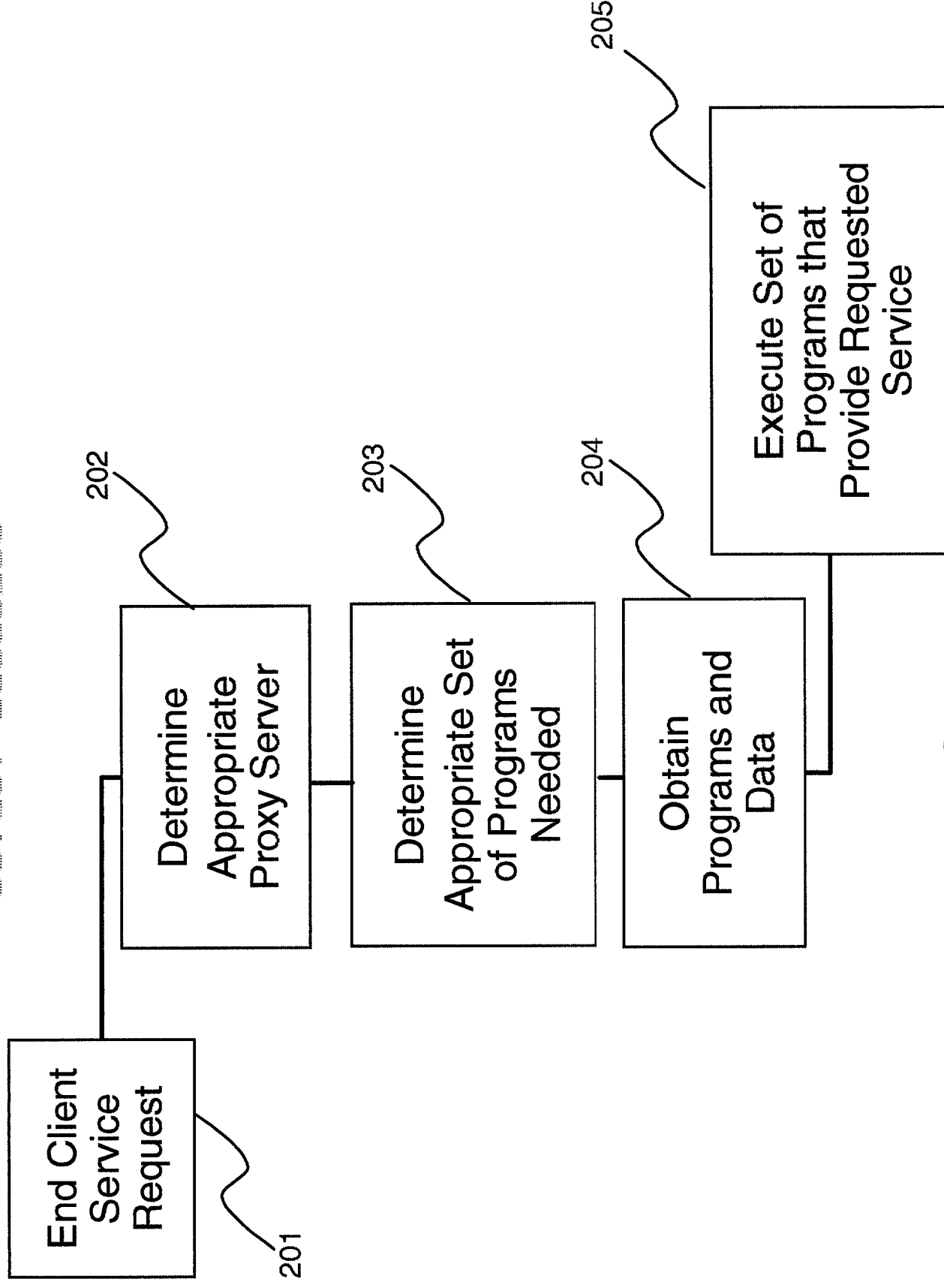


FIG 2

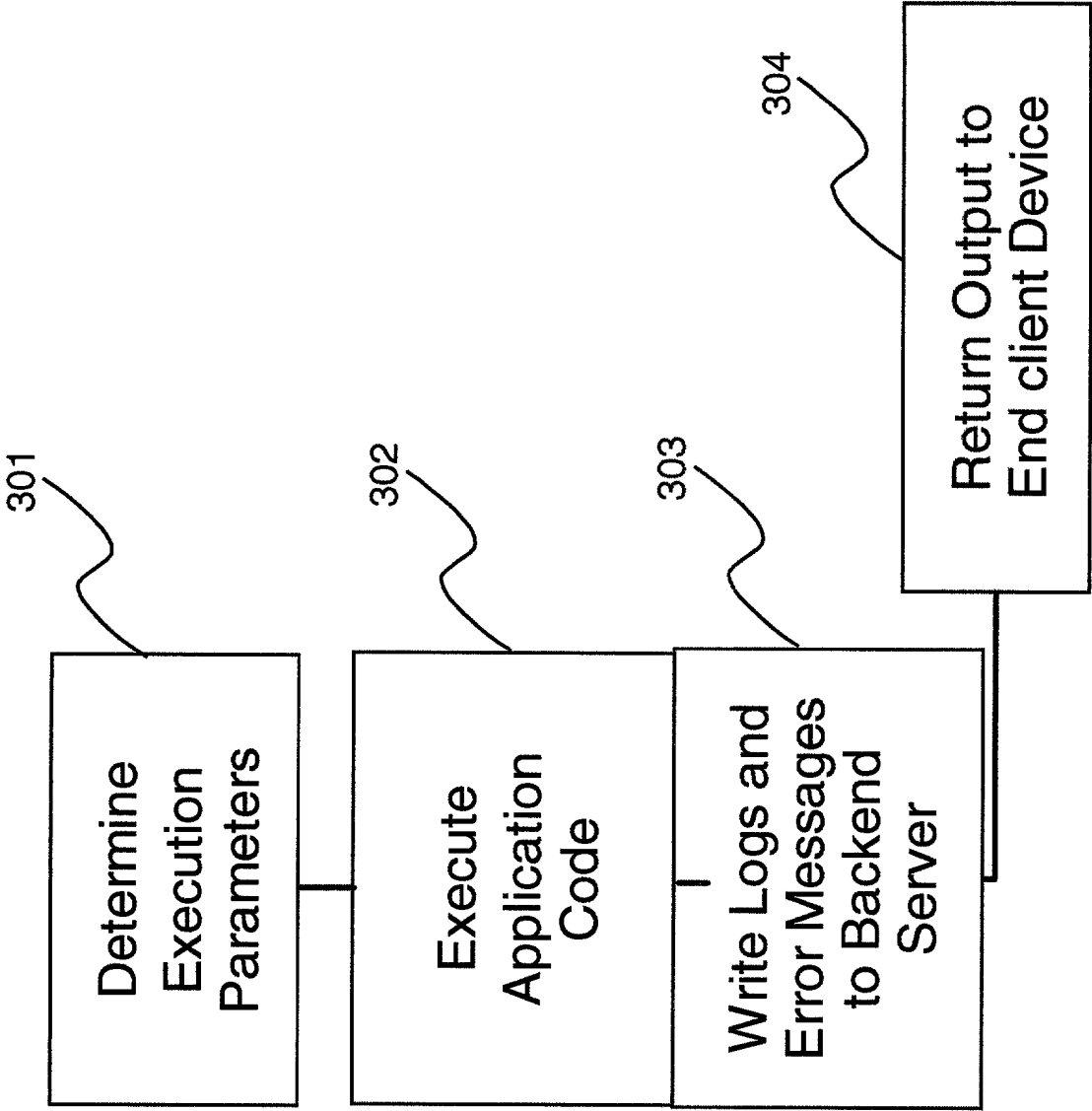


FIG 3

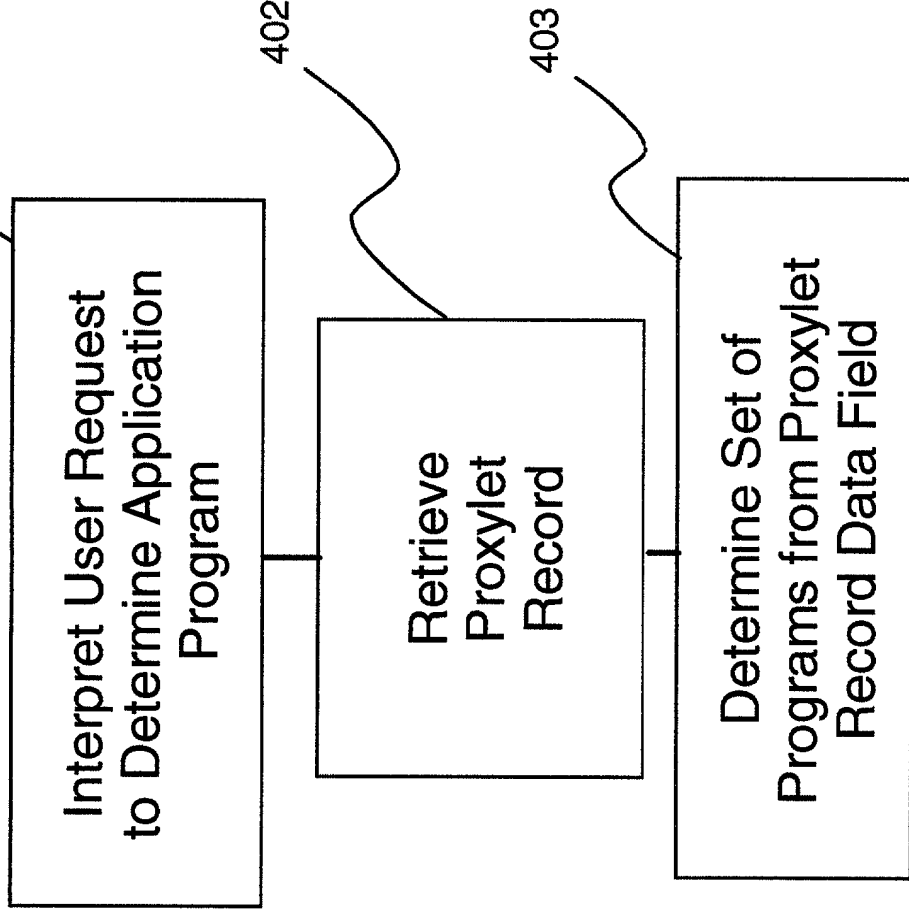


FIG 4

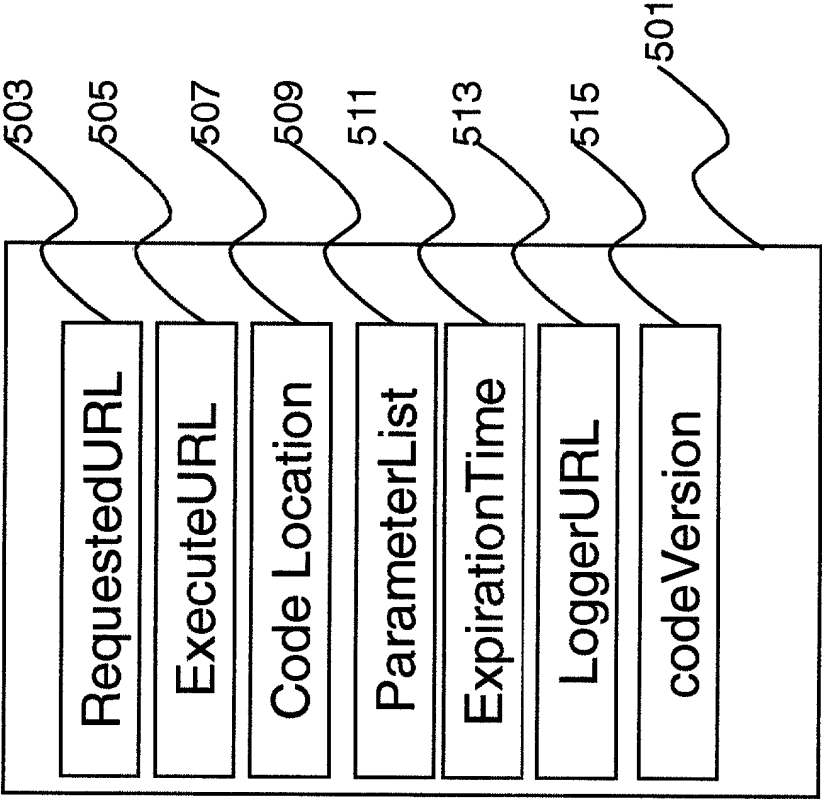


FIG 5

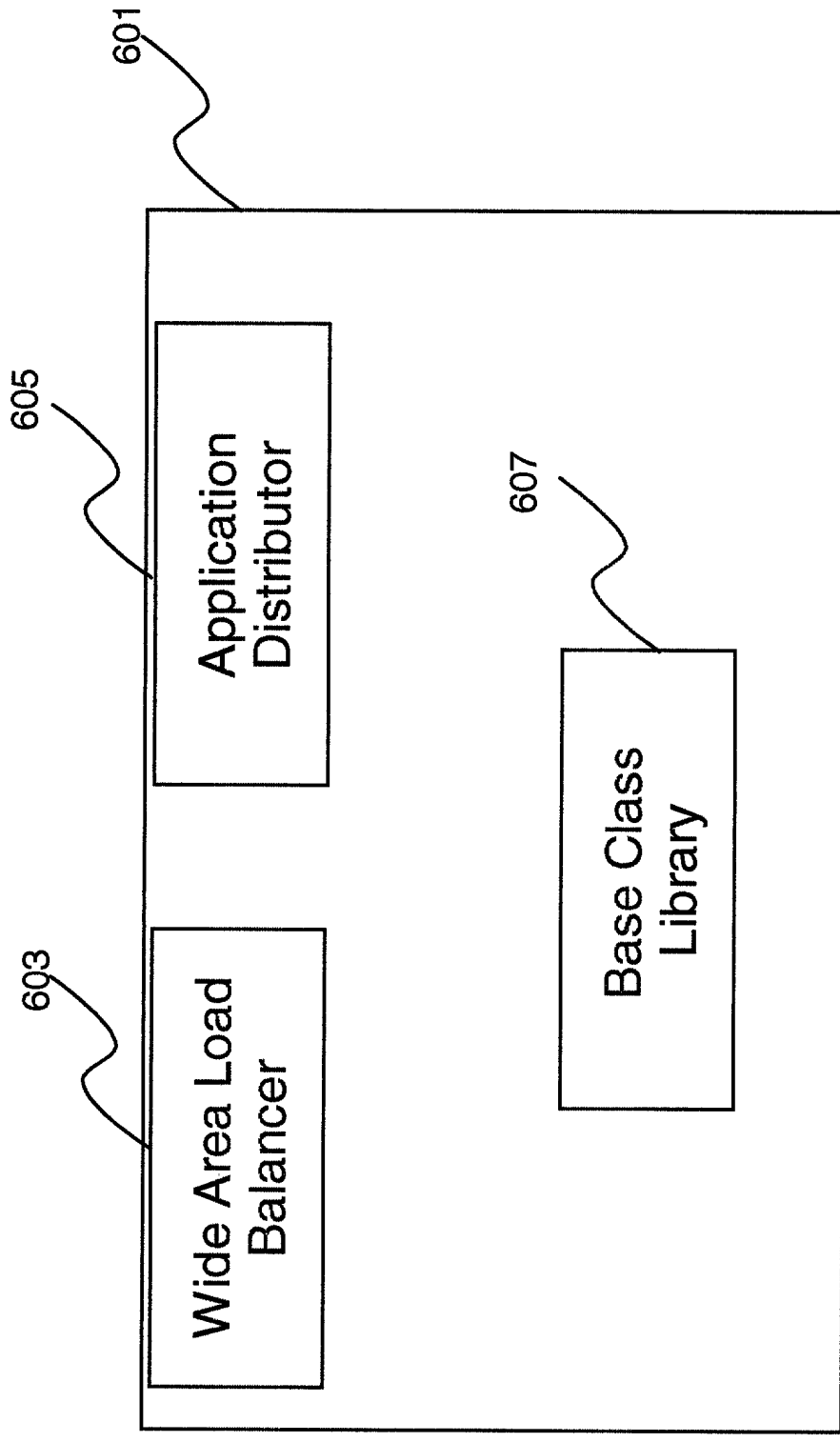


FIG 6

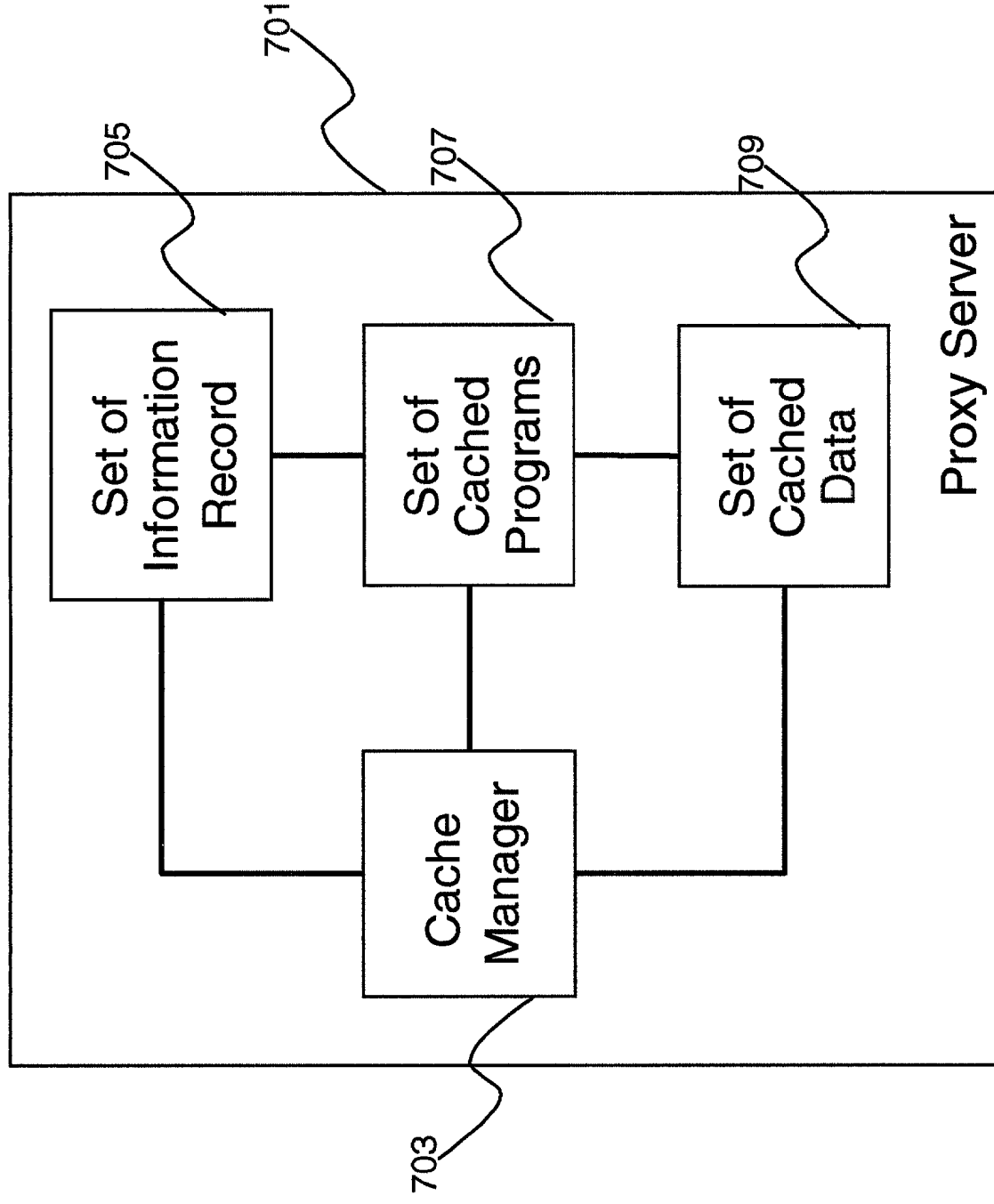


FIG. 7

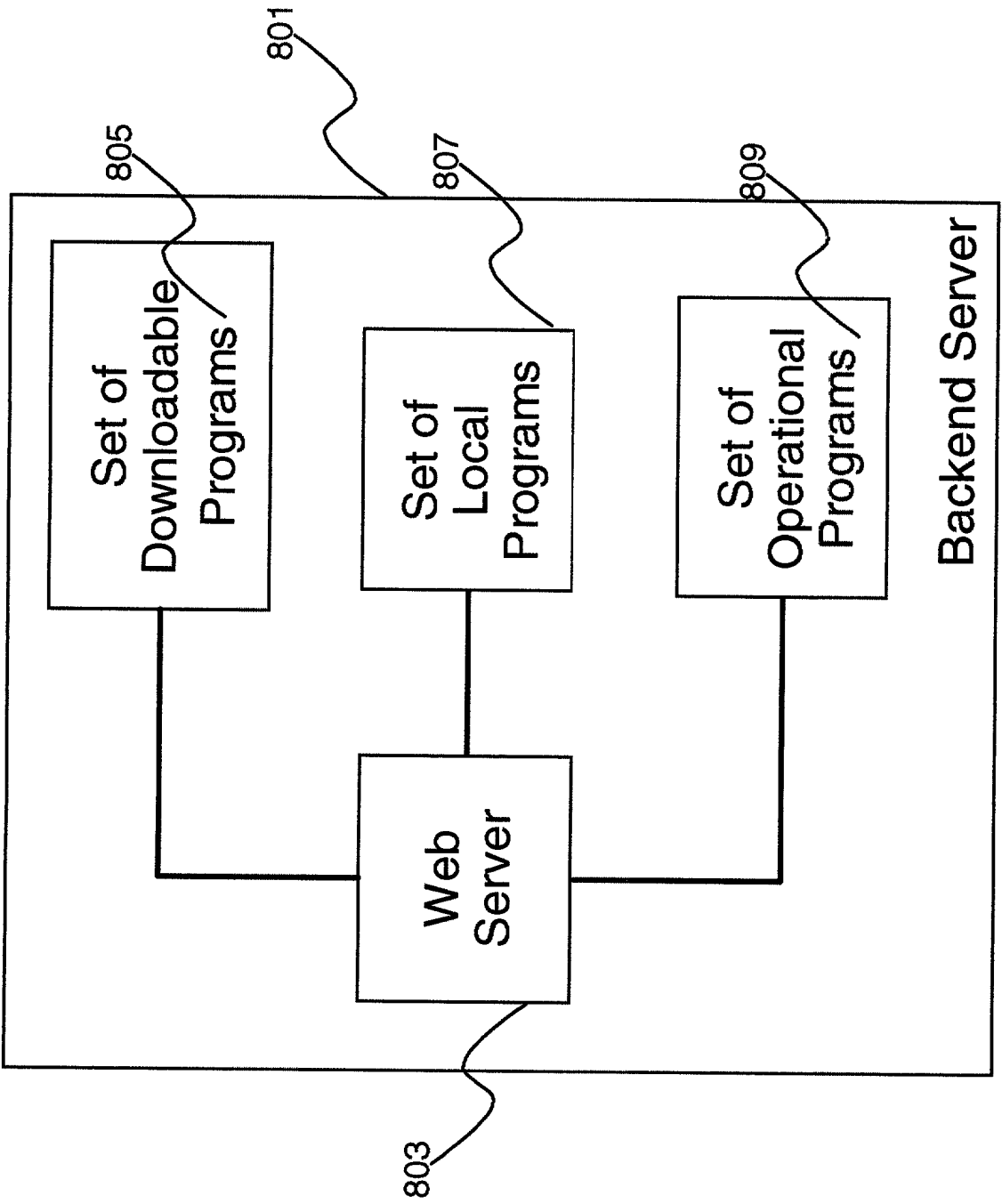


FIG. 8